LONG RANGE NAVIGATION FOR MARS ROVERS USING SENSOR-BASED PATH PLANNING AND VISUAL LOCALISATION

Sharon L. Laubach, Clark F. Olson, Joel W. Burdick, Samad Hayati

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

ABSTRACT

The Mars Pathfinder mission illustrated the benefits of including a mobile robotic explorer on a planetary mission. However, for future Mars rover missions, significantly increased autonomy in navigation is required in order to meet demanding mission criteria. To address these requirements, we have developed new path planning and localisation capabilities that allow a rover to navigate robustly to a distant landmark. These algorithms have been implemented on the JPL Rocky 7 prototype microrover and have been tested extensively in the JPL MarsYard, as well as in natural terrain.

1. INTRODUCTION

Mars sample return missions currently being planned call for rovers capable of operation for up to a year. The rovers are required to traverse up to 100m/sol and to reach ground-specified targets accurately. Lessons learned from Mars Pathfinder indicate a need for significantly increased rover autonomy in order to meet mission criteria within severe constraints including limited communication opportunities with Earth, power, and computational resources. Each rover will be working in unknown, rough terrain. Given a goal that cannot be seen from the rover's location, the rover must use its sensors to navigate safely and accurately to the goal using autonomous processes. This will require, in particular, improved motion planning and localisation algorithms.

To address the constraints upon motion planning for Mars rovers, we have developed the *RoverBug* algorithm, which can be considered a "sensorised" version of the classical Tangent Graph (or "reduced visibility graph" [2]) concept. The RoverBug algorithm uses two operational modes, *motion-to-goal* and *boundary following*, which interact to ensure global convergence. In addition, a "virtual" submode of boundary following improves efficiency and handles the limited field-of-view (FOV). Motion-to-goal is typically the dominant behaviour. It directs the robot to move towards the goal using a local version of the tangent graph, restricted to the visible region. After executing the resultant subpath, motion-to-goal begins anew. This behaviour is continued until the goal is reached or the robot encounters a blocking obstacle. In the latter case, the planner switches to the boundary following behaviour.

The objective of the boundary following mode is to skirt the boundary of the obstacle, finding shortcuts where possible. Upon first detecting the blocking obstacle, the algorithm "virtually slides" along the obstacle boundary using gaze control, avoiding unnecessary motion toward the obstacle. Boundary following continues until the robot either completes a loop, in which case the goal is unreachable and the algorithm halts, or the locally visible region contains a new subpath toward the goal. In the latter case, the mode switches back to motion-to-goal. It can be shown that with these two operational modes working together, the RoverBug algorithm is guaranteed to reach the goal (or halt if the goal is unreachable) in finite time, is correct, and produces locally optimal (shortest-length) paths. Furthermore, RoverBug deals with the limited FOV of flight rovers in a manner which is efficient and minimises the need to sense and store data, using autonomous gaze control.

A complementary rover localisation algorithm is used to determine the change in the rover position by comparing terrain maps generated before and after each subpath is traversed. While the rover plans its movements, the terrain sensed by the rover cameras is compiled into a digital elevation map. After traversing the subpath generated by the planner, the rover senses the terrain through which it has just moved and generates a second terrain map that is registered to the first in order to determine the change in the

rover position. The map registration is performed by determining the relative position that optimises a maximum-likelihood similarity measure. An efficient multi-resolution search is used to determine the optimal registration without examining each position explicitly. By fitting the likelihood function that is computed with a parameterised surface, we compute subpixel localisation estimates. In addition, the uncertainty in the localisation can be estimated in order to combine the result with other sensors, for example using an extended Kalman filter.

Both RoverBug and the localisation algorithm have been implemented on the JPL Rocky 7 prototype microrover, a research vehicle designed to test technologies for future missions. Rocky 7, which is roughly the same size as the Sojourner rover now on Mars, has three stereo pairs of cameras for navigation: two body-mounted, and one on a deployable 1.2m mast. The implementation has been tested in the JPL MarsYard as well as in natural arroyo terrain, including traverses for tens of meters requiring multiple iterations of the motion planning and localisation algorithms. Together, these algorithms significantly augment microrovers' autonomous navigation ability, which in turn will aid in producing successful mobile robot missions.

2. PATH PLANNING

The current scenario for a rover sensing system consists of a stereo pair of cameras mounted on a mast, as well as two body-mounted stereo pairs, fore and aft. Typically, the mast cameras have a 30° to 45° field of view (FOV) and the body-mounted cameras an 80° to 100° FOV, and the "visible region" connected with these sensors sweeps out roughly a wedge, with limited downrange radius. On Rocky 7, stereo triangulation is used to generate a wedge-shaped terrain map [4]. A step/slope model [5] is used to detect obstacle pixels within this range image, and the convex hulls of distinct obstacles are computed. Next, the system "grows" the obstacles' convex hulls, accounting for the size of the rover as well as incorporating an empirically-determined safety buffer, to create the configuration space obstacles, or "C-obstacles." (See Fig.2 for an example, in this case using multiple stereo images to form a single combined "wedge" view.) If the goal lies within a C-obstacle, the obstacle's vertices are marked as goals, so an operator can designate a particular rock as a target, e.g., for later instrument placement. Each C-obstacle vertex is also labelled if it lies within another obstacle, or outside the boundaries of the current wedge.

Figure 1: The Rocky7 Prototype Microrover, developed at JPL to test technologies for future missions. It is pictured here in the JPL MarsYard, an outdoor testing arena featuring simulated martian terrain.

Due to severe constraints on computational resources, the RoverBug motion planner is designed to identify the minimal number of sensor scans needed—and which specific areas to scan—to proceed at each step, while avoiding unnecessary rover motion. The planner, based upon the Wedgebug algorithm developed in [3], uses a streamlined local model (the C-obstacles) which is renewed at every step, thus avoiding the issues of maintaining a global map, which taxes the limited memory available and is sensitive to registration errors. However, the algorithm does require good localisation to track the goal position and to determine whether the rover has executed a loop around an obstacle. Hence, the planner has been paired with the on-board localisation algorithm described in Section 4.

The RoverBug algorithm relies upon the construction of the local version of the tangent graph within the visible "wedge." The tangent graph consists of all line segments in freespace connecting the initial position, the goal, and all obstacle vertices, such that the segments are tangent to any obstacles they encounter. Let LTG(S) be the *local tangent graph* within the set S, defined as the tangent graph restricted to S.

The next two subsections describe the operational modes of the RoverBug algorithm in more detail. (Of note, no information, other than explicitly recorded points and parameters, is passed between steps.)

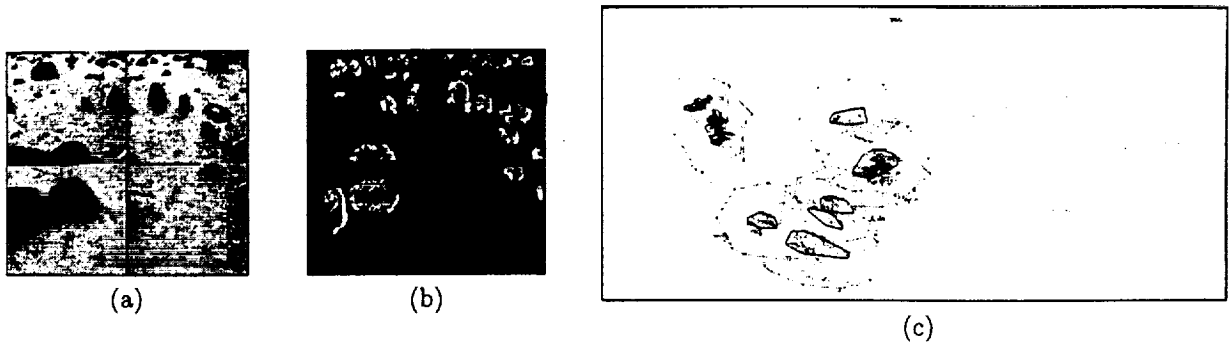

Figure 2: Results from a multi-image "wedge" view. (a) Left images from the mast-mounted stereo pair (b) Height of pixels determined using stereo triangulation (Black pixels indicate no data) (c) Overhead view of elevation map, with detected obstacles' convex hulls and corresponding C-obstacles, and a computed subpath

2.1. MOTION-TO-GOAL

The basic thrust of the motion-to-goal mode is monotonic progress toward the goal. At the beginning of the path sequence, an initialisation step records the parameter $d_{LEAVE} = d(A, T)$, where A is the rover's initial position, and T is the goal. This parameter marks the largest distance the rover can stray from T during a motion-to-goal segment. A motion segment is composed of a series of steps, consisting generally of a sensing, a planning, and then an execution phase, within a single operational mode.

Each motion-to-goal step proceeds as follows: The rover (at position x) first senses a wedge, $W_0 = W(x, \vec{v}_0)$, where $\vec{v}_0 = \overrightarrow{xT}$ is the vector from x to the goal, and constructs LTG(W_0). The LTG nodes comprise the convex C-obstacle vertices, the current rover position, and an optional node T_g in the direction of the goal. (Only those vertices within the visible region and on the exterior of the set of C-obstacles are used.) If there are no visible obstacles directly between the rover and the goal, T_g is added so the LTG contains a path directly towards T. The planner searches a subgraph of the LTG, $G1(W_0) = \{V \in \text{LTG}(W_0) | \, d(V, T) \leq \min(d(x, T), d_{LEAVE})\}$, for the locally optimal (shortest length) path to the goal, using an A* graph search method.

If a path is found, a subpath is generated by truncating the path at the far radius of the visible wedge (leaving an empirical buffer so the rover does not begin the next step directly behind a previously unsensed obstacle), and the planner returns the LTG nodes along the subpath (and the point where the path was truncated) as waypoints for the path execution algorithm.

This cycle repeats until either the rover reaches the goal, or no clear path to T exists within the visible region. If the planner detects that the rover cannot make forward progress through the current wedge, the rover must skirt a *blocking obstacle* to reach the goal. RoverBug then switches to its boundary following mode.

2.2. BOUNDARY FOLLOWING

Upon detecting a blocking obstacle O, it is clear that the rover must circumvent the obstacle in order to resume progress toward T. Unfortunately, the Rocky 7 mast is not capable of detecting obstacles reliably within roughly 1m of the vehicle, so boundary following must be accomplished using the body-mounted stereo pairs. These cameras have a limited useful range, roughly 0-1.5m from the vehicle, and cannot generally "see behind" obstacles (as can the mast cameras). Therefore, being close to an obstacle restricts the rover's already-limited view and can result in tiny incremental steps. In order to efficiently acquire data from the robot's current position and to avoid as much inefficient motion as possible, we add a submode of boundary following, called "virtual boundary following."

In essence, the object of "virtual boundary following" is to swing the mast cameras back and forth in a prescribed manner, to search for the "best" place to move and begin "normal boundary following," thus generating a local shortcut in the rover's path. First, the planner chooses a temporary "positive" sense of rotation by selecting the side of the blocking obstacle with the shortest path to T (which will pass outside of the visible region). Next, the rover scans the wedge $W_1 = W(x, \vec{v}_1)$, where $\angle(\overrightarrow{xT}, \vec{v}_k) = 2k\alpha_{mast}$ (α_{mast} is the half-angle subsumed by a mast wedge).

Let $\overline{W} = \bigcup^{sensed} W_k(x)$. The planner computes LTG($\overline{W}$). We define the wedge boundary as the two rays bounding the visible region; the arc defining the downrange radius is considered interior to the wedge. If $\exists$ a node $V \in$ LTG($\overline{W}$) $\cap\ \partial O$ such that $V \in$ int($\overline{W}$), the robot moves to V and begins "normal boundary following," first recording two features: d_{reach}, the closest distance to T encountered so far on ∂O, and V_{loop}, the intersection of (the near side of) ∂O with the bounding ray in the "negative" direction. If there is no such node V, the planner directs the sensor to scan $W_{-1} = W(x, \vec{v}_{-1})$, constructs $\overline{W} = W_0 \cup W_1 \cup W_{-1}$, and searches the freshly expanded LTG($\overline{W}$). In this manner, the robot scans back and forth until a suitable node is found, then travels there to begin "normal boundary following."

"Virtual boundary following" ends when one of two events are detected:

1. $\exists V \in$ LTG($\overline{W}$) $\cap\ \partial O$ such that $V \in$ int($\overline{W}$). The robot moves to V, and begins normal boundary following.

2. The latest scanned wedge overlaps a previously scanned region (i.e., $|\angle(\vec{v}_0, \vec{v}_{last})| > \pi$). In this case, the robot is trapped by an encircling obstacle, and the algorithm halts.

"Normal boundary following" uses two views, one toward the goal and one in the direction of travel around the obstacle boundary, to determine whether a clear path towards the goal exists while the robot circumnavigates the obstacle. In this mode, at the start of each step, the rover turns toward the goal and uses its body-mounted cameras to sense W_0, then searches $G1(W_0)$. If $T \in W_0$, the rover moves to T and the algorithm halts. Otherwise, if there is a clear path to T through W_0, the planner directs the rover to raise its mast and image toward the goal. (Rocky 7 is unable to have its mast deployed as it moves.) Boundary following exits here if $\exists V \in G1(W_0)$ such that $d(V,T) < d_{reach}$, the *leaving condition*, in which case the planner resets d_{LEAVE} to $d(V,T)$, and begins a new motion-to-goal segment.

If neither of these conditions hold, the rover turns in the positive direction by α_{body} (the half-angle subsumed by the body-mounted cameras), senses a new wedge, and constructs the new conglomerate wedge $\overline{W}$. If $V_{loop} \in W(x, \vec{t}_x)$, and $V_{loop} \in$ the connected portion of ∂O containing x, the robot has executed a loop—therefore, the goal is unreachable, and the algorithm halts. Otherwise, the planner computes $V \in \partial O \cap$ LTG($W(x, \vec{t}_x)$) such that $d(x,V) > d(x,V')$ $\forall V' \in \partial O \cap$ LTG($W(x, \vec{t}_x)$). If $V \in int\overline{W}$, the robot records d_{reach}, executes this subpath, then begins a new boundary following step. Otherwise, the rover turns again. The rover stops turning either when it has detected V_{loop}, has found a suitable point V, or has turned so far that it is overlapping an area already contained in $\overline{W}$, in which case the algorithm aborts.

3. TRAVERSE

The execution of each subpath in the implementation of this system on Rocky 7 is accomplished using the "Go-to-Waypoint" algorithm described in [7] as a heuristic collision avoidance mechanism. Future work will incorporate a path-execution algorithm designed to follow a series of waypoints, discarding those passed during collision avoidance manoeuvers, and able to request a replan if the rover strays too far from its computed path.

4. LOCALISATION

After a subpath has been traversed, localisation is performed in order to correct errors in dead-reckoning that have accumulated. This is accomplished by imaging there terrain through which the rover has just moved and comparing it to the map generated prior to the path planning for this subpath. Both terrain maps are generated using stereo vision on-board the rover [4].

4.1. MAP SIMILARITY MEASURE

In order to formulate the matching problem in terms of maximum-likelihood estimation, we use a set of measurements that are a function of the robot position. A convenient set of measurements are the distances from the occupied cells in the local map to their closest occupied cells in the global map. Denote these distances $D_1^X, ..., D_n^X$ for the robot position X. The likelihood function for the robot position can be formulated as the product of the probability densities of these distances. For convenience, we work in the $\ln L(X)$ domain:

$$\ln L(X) = \sum_{i=1}^{n} \ln p(D_i^X)$$

For the uncertainty estimation to be accurate, it is important that we use a probability density function (PDF) that closely models the sensor uncertainty. This can be accomplished using a PDF that is the weighted sum of two terms:

$$p(D_i^X) = \alpha p_1(D_i^X) + (1-\alpha)p_2(D_i^X)$$

The first term describes the error distribution when the cell is an inlier (in the sense that the terrain position under consideration in the local map also exists in the global map). In this case, D_i^X is a combination of the errors in the local and global maps at this position. In the absence of additional information with respect to the sensor error, we approximate $p_1(D_i^X)$ as a normal distribution:

$$p_1(D_i^X) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(D_i^X)^2/2\sigma^2}$$

The second term describes the error distribution when the cell is an outlier. In this case the position represented by the cell in the local map does not appear in the global map. This may be due to range shadows that were present when the global map was constructed or outliers that are present in the range data when the local map is constructed. In theory, this term should also decrease as D_i^X increases, since even true outliers are likely to be near some occupied cell in the global map. However, this allows pathological cases to have an undue effect on the likelihood for a particular robot position. In practice, we have found that modeling this term as a constant is both convenient and effective:

$$p_2(D_i^X) = K$$

4.2. SEARCH STRATEGY

A multi-resolution search strategy is used to determine the most likely robot position [1, 6]. This method is guaranteed to locate the optimal position in the discretised search space. The pose space is first discretised at the same resolution as the occupancy grids so that neighboring positions in the pose space move the relative positions of the grids by one grid cell. We then test the nominal position of the robot given by dead-reckoning so that we have an initial position and likelihood to compare against. Next, the pose space is divided into rectilinear cells. Each cell is tested to determine whether it could contain a position that is better than the best position found so far. Cells that cannot be pruned are divided into smaller cells, which are examined recursively. When a cell is reached that contains a single position in the discretised pose space, then this position is tested explicitly.

To determine whether a cell C could contain a pose superior to the best found so far, we examine the pose c at the center of the cell. A bound is computed on the maximum distance between the location to which a cell in the local map is transformed by c and by any other pose in the cell. We call this distance Δ_C. For the space of translations, Δ_C is simply the distance between c and any corner of the cell. To place a bound on the quality of any position within the cell, we bound each of the distances that can be achieved by features in the local map over the cell. This is done by subtracting the maximum change of the cell, Δ_C, from the distance achieved at the center of the cell, D_i^c:

$$D_i^C = \max(D_i^c - \Delta_C, 0)$$

The values obtained are then propagated through the likelihood function to bound the score that can be achieved by any position in the cell.

$$P_i^C = \ln p(D_i^C)$$

P_i^C is now the maximum score that the ith feature of the local map can contribute to the likelihood for any position in the cell.

A bound on the best overall likelihood that can be found at a position in the cell is given by:

$$\max_{X \in C} \ln L(X) \leq \sum_{i=1}^{n} P_i^C$$

If this bound does not surpass the best that we have found so far, then the entire cell is pruned from the search. Otherwise, the cell is divided into two cells by slicing it along the longest axis and the process is repeated recursively on the subcells.

4.3. SUBPIXEL LOCALISATION

Using this probabilistic formulation of the localisation problem, we can estimate the uncertainty in the localisation in terms of both the variance of the estimated positions and the probability that a qualitative failure has occurred. Since the likelihood function measures the probability that each position in the pose space is the actual robot position, the uncertainty in the localisation is measured by the rate at which the likelihood function falls off from the peak. In addition, we can perform subpixel localisation in the discretised pose space by fitting a surface to the peak that occurs at the most likely robot position.

We assume that the likelihood function can be approximated as a normal distribution in the neighborhood around the peak location. Fitting such a normal distribution to the computed likelihoods yields both an estimated variance in the localisation estimate and a subpixel estimate of the peak location. While the

approximation of the likelihood function as a normal distribution may not always be ideal, it yields a good fit to the local neighborhood around the peak and our experimental results indicate that very accurate results can be achieved under this assumption.

In addition to estimating the uncertainty in the localisation estimate, we can use the likelihood scores to estimate the probability of a failure to detect the correct position of the robot. This is particularly useful when the terrain yields few landmarks or other references for localisation and thus many positions appear similar to the robot.

4.4. TARGET SELECTION

Prior to performing localisation, the rover analyses the terrain in the map generated at the initial rover position in order to select a *localisation target*. This target is the position in the terrain that the rover looks at in order to generate a new map to match against the previously generated map. We want to select a location that has very distinctive terrain and that allows the localisation to be performed with the smallest uncertainty.

The localisation target is determined by estimating the amount of error present in the map computed at the initial rover position as well as the amount of error that would be generated by imaging the terrain from the final rover position. These errors are encoded in a probability map of the terrain expected to be seen from the final rover position. Each cell in this map contains an estimate of the probability that the cell will be seen as occupied by the rover. By treating this probability map as a terrain map and comparing it to the map generated at the initial rover position, we can predict the uncertainty that will occur in the localisation for any target that the rover may look at to use for terrain matching. The location with the lowest predicted uncertainty is selected as the localisation target.

5. RESULTS

The implementation of the RoverBug and localisation algorithms on the Rocky 7 prototype Mars rover has been tested in the JPL MarsYard and in natural terrain, for traverses up to tens of meters requiring several iterations of both algorithms. The basic scenario is as follows: the rover is situated in unknown, rough terrain. The remote human operator designates a goal, which is generally outside the range of the rover's sensors, and sets in motion the autonomous navigation system. The system begins by directing the mast to image towards the goal, generating data for both the localisation and motion planning algorithms. The RoverBug algorithm searches the resulting LTG, and directs the mast to look in the appropriate direction(s) to produce the first subpath. Upon the traversal of the first subpath, the localisation algorithm corrects the rover's position estimate. The cycle repeats, and the system incrementally builds and executes each subpath until the goal is reached.

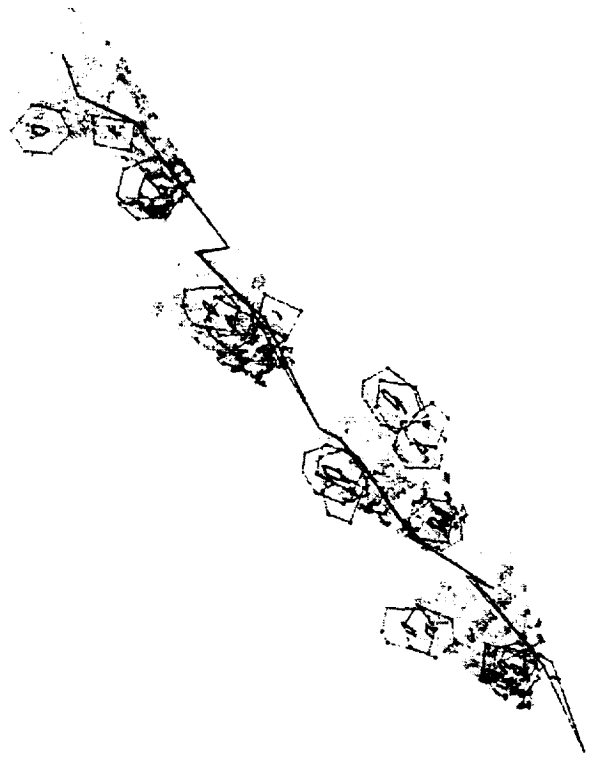

Figure 3: Results from a multi-step run in the JPL MarsYard. The path begins in the lower right corner of the image, toward a goal approx. 21m distant in the upper left. Each (single-image) wedge depicts a rangemap produced from mast imagery, and extends roughly 5m from the imaging position. The obstacles are marked by a black convex hull, and a grey C-obstacle. Each subpath ends with an apparent "jag" in the path; these are not in fact motions, but rather the result of the localisation procedure run at the conclusion of each step. The second line echoing the path is the rover's telemetry for the run.

Fig. 2 demonstrates the generation of a path segment from a multiple mast images, treated as a single "wedge" view. The generated path skirts all of the obstacles and achieves the goal using data from all four of the stereo pairs.

Fig. 3 shows the results of one typical run in the MarsYard. The goal was approximately 21m distant from the initial position, and the radius of each wedge was 5m. The obstacles' convex hulls and silhouettes are computed within each wedge view, and a subpath generated, which is executed before the next wedge view is taken. The steps of the localisation algorithm straddle each path-planning cycle, generating an updated position estimate after the execution of each subpath. The resultant multi-step path runs from lower right to upper left.

6. SUMMARY

The specifications for autonomous rovers for the currently planned Mars missions place strenuous requirements on the rovers' ability to traverse long distances to ground-specified targets safely and accurately. A system able to achieve accurate long-range navigation through planetary terrain is described, which combines sensor-based motion planning and visual localisation. Results from the Rocky 7 prototype rover are presented, which demonstrate good performance of the system.

ACKNOWLEDGEMENTS

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work is an element of the Long Range Science Rover project, which is developing technology for future Mars missions. This project is funded as part of the NASA Space Telerobotics Program, through the JPL Robotics and Mars Exploration Technology Office.

References

[1] D. P. Huttenlocher and W. J. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 705–706, 1993.

[2] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[3] S. L. Laubach, and J. W. Burdick. An autonomous sensor-based path-planner for planetary microrovers. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1999.

[4] L. Matthies. Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation. *International Journal of Computer Vision*, 8(1):71–91, July 1992.

[5] L. Matthies, A. Kelly, T. Litwin, and G. Tharp. Obstacle detection for unmanned ground vehicles: A progress report. In *Proceedings of the 1995 Intelligent Vehicles Symposium*, pages 66–71. 1995.

[6] C. F. Olson and L. H. Matthies. Maximum-likelihood rover localization by matching range maps. In *Proceedings of the International Conference on Robotics and Automation*, pages 272–277, 1998.

[7] R. Volpe, J. Balaram, T. Ohm, and R. Ivlev, The Rocky7 Mars rover prototype. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1996.